PoseMonitor 使用說明

📖 這是什麼？

`PoseMonitor` 是一個**監控工具**，用來追蹤：

1. 機器人手臂末端（夾爪）的位置和姿態
2. 夾爪和目標物體（例如風扇）之間的距離和角度誤差
3. 夾爪是否成功夾住物體

🚀 快速開始

使用create_default 快速鍵立 monitor必須要有3個prim paths和機器人相關的檔案位置：

```
from pose_monitor import PoseMonitor

# 用 factory method 一行搞定
monitor = PoseMonitor.create_default(
    robot_prim_path="/World/WorkSpace/RS_M90E7A_Left",       # 機器人的 USD 路徑
    fan_prim_path="/World/WorkSpace/Scene/Fan",               # 風扇的 USD 路徑
    ground_truth_prim_path="/World/WorkSpace/Scene/GroundTruth",  # 目標位置的 USD 路徑
    robot_description_path="path/to/robot_description.yaml", # 機器人描述檔
    urdf_path="path/to/robot.urdf",                           # 機器人 URDF 檔
```

```
)

# ⚠️ 重要！模擬開始後一定要呼叫 initialize()
monitor.initialize()

# 現在可以開始使用了！
```

📌 常用功能

1. 取得夾爪目前的位置和姿態

```
ee_pose = monitor.get_end_effector_pose()

print(f"夾爪位置: {ee_pose.p}")       # 輸出: [x, y, z] 三維座標
print(f"夾爪姿態: {ee_pose.q}")       # 輸出: [w, x, y, z] 四元數
```

2. 計算夾爪和風扇之間的誤差

```
error = monitor.get_ee_to_fan_error()

print(f"距離: {error.distance:.3f} 公尺")
print(f"角度誤差: {np.degrees(error.angle_error):.1f} 度")
```

回傳的 `PoseError` 物件包含：

屬性	說明
`distance`	直線距離（公尺）
`position_error`	位置差向量 `[dx, dy, dz]`
`angle_error`	旋轉角度誤差（弧度）
`rotation_error`	3x3 旋轉矩陣

3. 檢查夾爪是否夾住風扇

```
if monitor.is_holding_fan():
    print("✅ 成功夾住風扇！")
else:
    print("❌ 還沒夾到")
```

抓取判定 (Grasp Logic)

Monitor 會檢查兩個條件，必須同時成立才算 `holding`：

1. **距離條件**：`grasp_zone_min` ≤ 距離 ≤ `grasp_zone_max`
2. **夾爪條件**：`grip_min` ≤ 夾爪滑塊位置 ≤ `grip_max`
   這兩個條件可在config之中改變其range。

4. 取得手臂關節角度

```python
# 取得 7 軸手臂的關節位置
arm_positions = monitor.get_arm_joint_positions()
print(f"手臂關節: {arm_positions}")  # 7 個數值的陣列

# 取得夾爪的開合程度
gripper_positions = monitor.get_gripper_joint_positions()
print(f"夾爪狀態: {gripper_positions}")  # [左手指, 右手指]
```

5. 取得手指和把手的距離（關節到把手TCP距離）

```python
left_dist, right_dist = monitor.get_finger_to_handle_distances()
print(f"左手指到左把手: {left_dist:.3f}m")
print(f"右手指到右把手: {right_dist:.3f}m")
```

把手的TCP是從風扇的TCP使用 offset推算的，此offset定義於config之中。

6. `get_finger_poses()`

取得左右手指在**世界座標系 (World Frame)** 中的實際姿態。這與 `get_end_effector_pose` 不同，後者是夾爪基座或 TCP 中心。 這邊取的點是以目前手指的關節位置為準，也就是slider9和slider10，而rotation則與夾爪TCP一致。

```python
left_finger, right_finger = monitor.get_finger_poses()

# left_finger 和 right_finger 都是 PosePq 物件
```

```
print(f"左手指世界座標: {left_finger.p}")
print(f"右手指世界座標: {right_finger.p}")
```

所有函數

類別	方法/屬性	功能簡述	回傳型別
Setup	`create_default(...)`	建立 Monitor (Factory)	`PoseMonitor`
	`initialize()`	初始化物理觀察者	`None`
Robot Pose	`get_end_effector_pose()`	取得夾爪中心姿態	`PosePq`
	`get_arm_joint_positions()`	取得手臂 7 軸角度	`np.ndarray`
	`get_gripper_joint_positions()`	取得夾爪 2 軸位置	`np.ndarray`
	`get_finger_poses()`	取得左右手指世界姿態	`(PosePq, PosePq)`
Object Pose	`get_handle_poses()`	取得左右把手世界姿態	`(PosePq, PosePq)`
Error/Dist	`get_ee_to_fan_error()`	夾爪與風扇的誤差	`PoseError`
	`get_ee_to_ground_truth_error()`	夾爪與 GT 的誤差	`PoseError`
	`get_pose_error_to_target(obj)`	夾爪與任意物件的誤差	`PoseError`
	`get_finger_to_handle_distances()`	手指到把手的個別距離	`(float, float)`
Logic	`is_holding_fan()`	判斷是否夾住	`bool`

關於 `PosePq` 資料結構

很多方法回傳 `PosePq`，它是一個簡單的 Data Class：

- `.p`: `np.ndarray` (shape: 3,) - 位置向量 $[x, y, z]$
- `.q`: `np.ndarray` (shape: 4,) - 四元數 $[w, x, y, z]$

🔧 參數設定

GraspDetectionConfig - 夾取偵測參數

以下是 `GraspDetectionConfig` 的所有參數：

參數名稱 (Attribute)	預設值	單位	說明與用途
抓取邏輯 (Grasp Logic)			用於 `is_holding_fan()` 判定
`grip_position_min`	`0.019`	m	**最小夾持閉合量** (絕對值)。 低於此值則視為夾空。 左指(slider9)需 $\geq$ 此值，右指(slider10)需 $\leq$ 負此值。
`grip_position_max`	`0.021`	m	**最大夾持閉合量** (絕對值)。
`grasp_zone_min_m`	`0.01415`	m	**最小有效距離**。 夾爪中心 (TCP) 與目標中心的最小距離。
`grasp_zone_max_m`	`0.02415`	m	**最大有效距離**。 超過此距離即使夾爪閉合，也會被視為夾空。

參數名稱 (Attribute)	預設值	單位	說明與用途
把手幾何 (Handle Geometry)			用於 `get_handle_poses()` 計算
`handle_y_offset`	`0.1`	m	**把手半寬**。 從物體中心沿著 Y 軸 (抓取軸) 到左右把手的距離。左把手為 $+Y$，右把手為 $-Y$。
`handle_x_offset`	`-0.015`	m	**前後偏移量**。 從物體中心沿著 X 軸 (接近軸) 的偏移。

ApproachFrameConfig - 目標座標系設定

以下是 `ApproachFrameConfig` 的參數（此設定包含在 `GraspDetectionConfig` 內）：

參數名稱	預設值	說明
`approach_axis`	`"+y"`	目標物體的哪個軸對應夾爪的 **+X 軸**（接近方向）
`grasp_axis`	`"-x"`	目標物體的哪個軸對應夾爪的 **+Y 軸**（夾取方向）

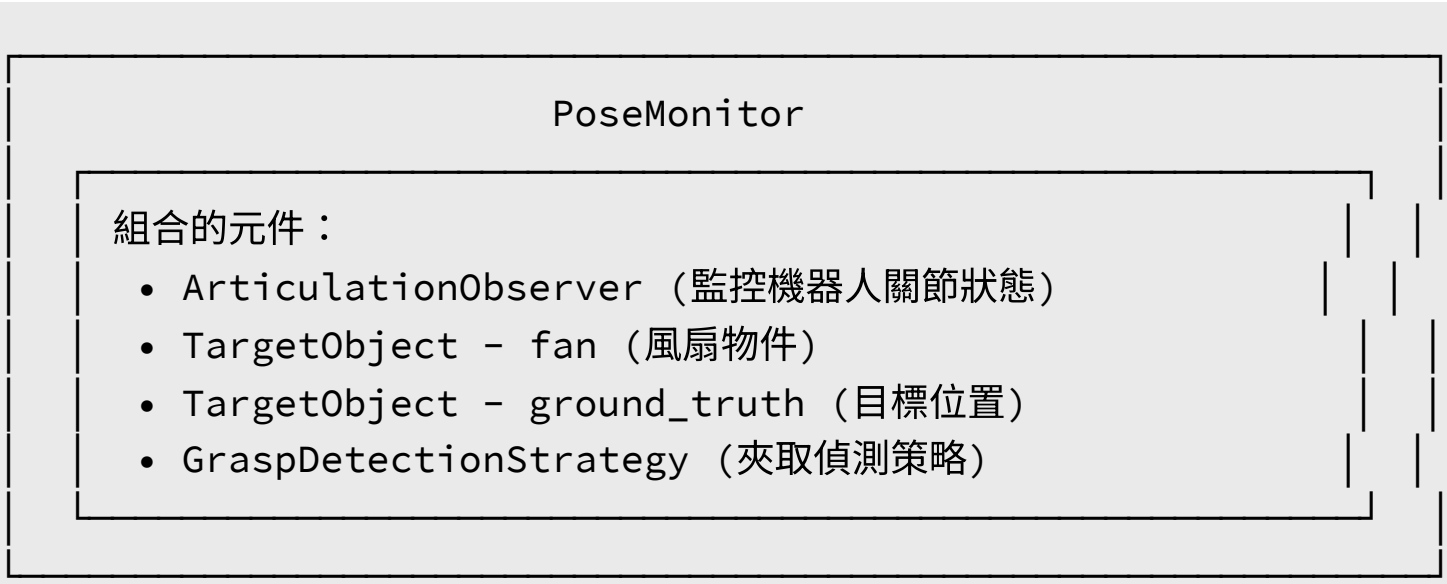
座標系對應說明：

這個設定用於將目標物體的 local 座標系轉換為夾爪（end effector）的座標系。

- **夾爪座標系慣例**：
  - **+X 軸**：接近方向（approach）- 夾爪向前移動的方向

  - **+Y 軸**：夾取方向（grasp）- 手指張開的方向
  - **+Z 軸**：上方向（由右手定則決定）
- **使用情境**：如果目標物體不是風扇，或座標系與預設不同時，需要設定這兩個參數。
- **可用值**：`"+x"`, `"-x"`, `"+y"`, `"-y"`, `"+z"`, `"-z"`

📦 類別總覽



什麼時候要呼叫 `initialize()` ？
A: 在 Isaac Sim 模擬開始**之後**、第一次使用 monitor **之前**呼叫。

📝 完整範例

```python
import numpy as np
from pose_monitor import PoseMonitor

# 建立 monitor
monitor = PoseMonitor.create_default(
    robot_prim_path="/World/WorkSpace/RS_M90E7A_Left",
    fan_prim_path="/World/WorkSpace/Scene/Fan",
    ground_truth_prim_path="/World/WorkSpace/Scene/GroundTruth",
    robot_description_path="project_asset/assets/RS-M90E7A/motion_policy_configs/rob
    urdf_path="project_asset/assets/RS-M90E7A/motion_policy_configs/RS-M90E7A.urdf",
)

# 模擬開始後初始化
monitor.initialize()

# 主迴圈中使用
def on_physics_step():
    # 取得目前誤差
    error = monitor.get_ee_to_fan_error()

    # 印出狀態
    print(f"距離目標: {error.distance:.3f}m, 角度誤差: {np.degrees(error.angle_error):.

    # 檢查是否夾住
    if monitor.is_holding_fan():
        print("夾住了！可以開始移動")
```

```
# 取得關節狀態（如果需要）
arm_joints = monitor.get_arm_joint_positions()
gripper_joints = monitor.get_gripper_joint_positions()
```

🔗 相關檔案

檔案	說明
`pose_monitor.py`	主要類別
`articulation_observer.py`	機器人關節狀態監控
`target_object.py`	目標物體 wrapper
`grasp_config.py`	設定類別（PosePq、GraspDetectionConfig 等）
